## REMARKS

In the Office Action, the Examiner issued a final rejection of Claims 1-18, which are all of the pending claims, over the prior art, principally U.S. Patent 6,247,172 (Dunn). More specifically, Claims 1, 2, 4, 6, 7 and 10-16 were rejected under 35 U.S.C. §102 as being fully anticipated by Dunn; and Claims 3, 5, 8, 9, 17 and 18 were rejected under 35 U.S.C. §103 as being unpatentable over Dunn in view of U.S. Patent 6,412,109 (Ghosh). In addition, Claims 3, 5, 17 and 18 were rejected under 35 U.S.C. §112 as being indefinite, and the Examiner objected to an informality in Claim 18.

Applicants herein ask that independent Claims 1, 6, 7, 8, 10, 12, 13, 15 and 16 be amended to better define the subject matters of these claims. Also, Claims 17 and 18 are being amended to correct informalities in these claims and to overcome the rejection of claims 17 and 18 under 35 U.S.C. §112.

For the reasons discussed below, Claims 1-18, as presented herewith, are clear and definite and patentably distinguish over the prior art. The Examiner is thus requested to enter this Amendment, and to reconsider and to withdraw the rejection of Claims 1-18 under 35 U.S.C. §§102 and 103. The Examiner is further asked to reconsider and to withdraw the rejection of Claims 3, 5, 17 and 18 under 35 U.S.C. §112 and the objection to Claim 18, and to allow Claims 1-18.

With respect to the rejection of Clams 3, 5, 17 and 18 under 35 U.S.C. §112, the Examiner rejected these claims as being indefinite for several reasons. First, the Examiner noted several informalities in Claims 17 and 18, and second, the Examiner argued that there is insufficient antecedent basis for the term "said exception handler" in Claims 3 and 5.

This opportunity is being taken to correct the informalities in Claims 17 and 18, as the Examiner suggested. For example, in Claim 17, "the post process" is being changed to "the post-processor," and in Claim 18, "the post processor also" is being changed to "the post-processor." In addition, in Claims 17 and 18, both occurrences of "the address" are being changed to "an address."

The second of the above-mentioned reasons for the Examiner's rejection of Claims 3, 5, 17 and 18 under 35 U.S.C. §112 is respectfully traversed. Claims 3 and 5 are dependent from Claim 1; and in Applicants' last Amendment, Claim 1 was amended to describe, in lines 7 and 8, "an exception handler." Thus, Claim 1 provides the appropriate antecedent basis for the term "said exception handler" in Claims 3 and 5.

In view of the foregoing, the Examiner is asked to enter the above-discussed changes to Claims 17 and 18, and to reconsider and to withdraw the objection to the language of Claims 17 and 18 and the rejection of Claims 3, 5, 17 and 18 under 35 U.S.C. §112.

With regard to the rejection of Claims 1-18 over the prior art, these claims patentably distinguish over the prior art because the references do not disclose or suggest the exception handling process as described in independent Claims 1, 6, 7, 8, 10, 12, 13, 15 and 16. In order to best understand this, it may be helpful if Applicants briefly review the present invention and the prior art.

The present invention generally relates to a procedure for transforming program code written for one machine into program code suitable for a second machine. More particularly, this invention provides effective optimization for such a program where that program may cause or include an exception process.

One challenge involved in emulation of code designed for one architecture (often a legacy platform) on a machine of a different architecture (referred to as the target machine or target platform) occurs in systems that support the handling of exceptions. When the conditions that would cause an exception in the original code occur, the translation system may need to ensure that the modeled state (the target machine state is consistent with the original, or legacy, machine state. This consistency is difficult to maintain in an aggressively optimizing translator. This is because optimization can re-arrange the order of execution of code, and when an exception occurs, the target machine state may not match the state expected by the old program.

The present invention effectively addresses this challenge by providing a program modification unit for modifying the object program in order to absorb a difference in content between the point of origin of an exception process, which occurs in response to the execution on the target machine of a command in the object program, and an exception handler at where the exception process is performed.

Dunn does not disclose or teach the program modification unit of this invention, and, more specifically, Dunn does not disclose or suggest compensating for the difference in the content of the code between the start of the exception process on the target machine and the exception handler.

In the Office Action, the Examiner argued that the recovery block 42 of Dunn corresponds to the program modification unit of the present invention. There are, though, important differences between the recovery block 42 of Dunn and the program modification unit of this invention.

More specifically, in this invention, the use of the program modification unit "to absorb a difference..." has a different objective from Dunn's recovery block 42. The machine state of the compensation code of this invention is the "target machine" state at an exception handler. However, the machine state of the recovery block of Dunn is the "legacy machine" state at the code that is throwing the current exception. Therefore, the program modification unit of this invention and the recovery block 42 of Dunn are at different locations and in different states. With the procedure described in Dunn, the recovery block contains instructions that complete all functions necessary to restore the target machine state to "the legacy machine state" (Dunn, column 3, line 13).

The legacy machine state is the machine state that could have resulted had the target platform executed "target code not optimized by the compiler" (Dunn, column 3, line 52). In contrast, with the procedure of this invention, program control is shifted, preferably through compensation codes, to the exception handler in order to "absorb the difference between the point of origin of the execution occurrence point on the target machine and the exception handler." The present invention, for instance, may perform the algorithms shown in Figures 4 to 7 in order to compensate a register image between the point of origin of the exception occurrence points on the target machine and the exception handler. With particular reference to Figure 10 of this application, Dunn's approach does not generate the "copy i and j variable values to R1 and R2" in the recovery block.

Independent Claims 1, 6, 7, 8, 10, 12, 13, 15 and 16 describe the above-discussed feature of this invention relating to the operation of the program modification unit. Claims 1 and 6 describe a program modification unit for modifying the object program in order to absorb a

difference in content between the point of origin of an exception process, which occurs in response to the execution on the target machine of a command in said object program, and an exception handler whereat said exception process is performed.

Claim 7 describes the step of preparing a basic block that includes a portion for examining, in an object program, a command that may cause an exception process on the target machine to determine whether an exception process has actually occurred, and that includes a command for, when an exception process has occurred on the target machine, moving program control to the exception handler. Claim 8, similarly, describes the step of establishing a basic block that includes a Try node for examining, in an object program a command that may cause an exception process on the target machine to determine whether an exception process has occurred, and a Catch node for performing an inherent process when an exception process has occurred on the target machine.

Claim 10 is directed to an optimization method for optimizing a program to increase processing efficiency on a target machine, and this claims positively sets forth the step of dividing software code, in an object program, that may cause an exception process on the target machine into software code for determining whether an exception process has occurred. Claim 10 describes the further step of designing a control flow graph so that when an exception process occurs on the target machine, program control is shifted to the code that actually caused the exception process.

Claims 12, 15 and 16 describe a process for, when a difference in content exists between the point of origin of an exception process on the target machine and an exception handler whereat said exception process is performed, generating in a basic block compensation code for

compensating for said difference. Analogously, Claim 13 describes a function for, when an exception process has occurred on a target machine, shifting program control to an exception handler whereat said exception process is run, and for, when a difference in content exists between the point of origin of said exception process on the target machine and said exception handler whereat said exception process is run, compensating for said difference before program control is shifted to said exception handler.

The other references of record have been reviewed. These other references, even when considered in combination, fail to disclose or suggest the above-discussed exception handling process. In particular, Ghosh was cited for its disclosure of try-catch block for handling exceptions. This reference, though, does not teach the use of an exception handler to identify a point in a process on the target machine with respect to which differences should be measured or compensated for, as described in Claims 1, 6, 7, 8, 10, 12, 13, 15 and 16.

Because of the above-discussed differences between Claims 1, 6, 7, 8, 10, 12, 13, 15 and 16 and the prior art, and because of the advantages associated with those differences, Claims 1, 6, 7, 8, 10, 12, 13, 15 and 16 patentably distinguish over the prior art and are allowable. Claims 2-5, 17 and 18 are dependent from Claim 1 and are allowable therewith. Similarly, Claim 9 is dependent from, and is allowable with, Claim 8; and Claims 11 and 14 are dependent from and are allowable with Claims 10 and 13 respectively.

It is noted that the changes asked for herein to Claims 1, 6, 7, 8, 10, 12, 13, 15 and 16 only more expressly describe features or processes already described in the claims. For instance, Claim 1 currently describes a program modification unit for modifying an object program, and the amendment requested herein only describes more specifically, how that modification occurs.

It is thus believed that entry of this Amendment is appropriate, and such entry is respectfully requested.

For the reasons set forth above, the Examiner is respectfully requested to enter this Amendment, to reconsider and to withdraw the objection to the language of Claim 18 and the rejections of Claims 3, 5, 17 and 18 under 35 U.S.C. §112. The Examiner is further asked to reconsider and to withdraw the rejection of Claims 1, 2, 4, 6, 7 and 10-16 under 35 U.S.C. §102 and the rejection of Claims 3, 5, 8, 9, 17 and 18 under 35 U.S.C. §103, and to allow Claims 1-18. If the Examiner believes that a telephone conference with Applicants' Attorneys would be advantageous to the disposition of this case, the Examiner is requested to telephone the undersigned.

Respectfully submitted,

John S. Sensny

John S. Sensny
Registration No. 28,757
Attorney for Applicants

Scully, Scott, Murphy & Presser
400 Garden City Plaza
Garden City, New York 11530
(516) 7472-4343

JSS:jy